Rapport TP1 de Métaheuristique en Optimisation

Edorh François (EDOF19059507), Guison Vianney (GUIV30069402)

17 février 2018

Table des matières

1 Méthodes implémentées

L'algorithme génétique est utilisable sur toute fonction à n arguments, avec $n >= 1$.

1.1 Crossovers

Toutes les méthodes de crossover sont dans le fichier Crossover.m.
Elles sont stockées dans la variable globale CROSSOVER.

1.1.1 Encodage binaire

Les variables réelles sont converties en entier entre 0 et $2^l - 1$, avec l la longueur du chromosome ($1 <= l <= 53$).
Chaque variable est représentée par un chromosome différent, chaque chromosome a la même longueur.

Les méthodes de crossover en encodage binaire suivantes ont été implémentées :

- single-point crossover,
- multi-point crossover, avec un paramètre de contrôle n compris entre 1 et $l - 1$,
- uniform crossover, avec deux variantes :
  - un paramètre p représentant une probabilité constante,
  - deux paramètres de contrôle p et t, où la probabilité d'une paire (a, b) est donnée par $P(T(a), T(b))$.

1.1.2 Encodage réel

Les méthodes de crossover sur valeurs réelles suivantes ont été implémentés :

- whole arithmetic crossover,
- local arithmetic crossover,
- blend crossover (ou $BLX-\alpha$), avec un paramètre de contrôle α (valeur par défaut de 0.5),
- simulated binary crossover, avec un paramètre de contrôle $n >= 0$.

1.1.3 Au choix

La méthode de crossover choisie est *1-Bit Adaptation*. C'est une méthode de crossover en encodage binaire.

Le principe est de définir deux méthodes de crossover différentes que l'on associe à la valeur 1 ou 0. Lors du crossover, on regarde le dernier bit des parents.
Si ils sont égaux, la méthode de crossover utilisée est celle associée à la valeur de ce bit (1 ou 0).
Si ils sont différents, un nombre aléatoire u entre 0 et 1 indique quelle méthode utiliser : si $u < 0.5$, on utilise la méthode associée à 0, sinon on utilise celle associée à 1.

Ajout personnel : étant donné que chaque variable possède un chromosome, avant la vérification du dernier bit des parents, un $\oplus$ (xor) entre les variables d'un même parent est effectué. C'est le résultat de ce calcul qui est comparé avec celui de l'autre parent.
$\oplus$ a été choisie car c'est la seule fonction binaire à deux opérandes qui produit 0 et 1 dans les mêmes proportions.

1.2 Mutations

Toutes les méthodes de mutation sont dans le fichier Mutation.m.
Elles sont stockées dans la variable globale MUTATION.

1.2.1 Encodage binaire

La méthode de mutation binaire implémentée est la mutation bit-flip.

1.2.2 Encodage réel

Les méthodes de mutation sur valeurs réelles suivantes ont été implémentées :

- uniform mutation,
- boundary mutation,
- normal mutation, avec un paramètre σ (nombre ou tableau),
- polynomial mutation, avec un paramètre de contrôle $n >= 0$,

- non-uniform mutation, avec un paramètre de contrôle b.

1.3 Sélections

Toutes les méthodes de sélection sont dans le fichier Sélection.m.
Elles sont stockées dans la variable globale SELECTION.

Les méthodes de sélection suivantes ont été implémentées :

- wheel selection,
- stochastic universal sampling,
- tournament selection, avec un paramètre de contrôle k, compris entre 1 et N, le nombre d'individus,
- unbiased tournament selection, avec un paramètre de contrôle k, compris entre 1 et N, le nombre d'individus,
- truncation selection, avec un paramètre c compris entre 1 et N, le nombre d'individus, où N/c correspond au nombre d'individus utilisés dans la sélection.

1.4 Faisabilité

Les deux méthodes permettant d'assurer la faisabilité d'une solution ont été implémentées dans Clamp.m.
Elles sont stockées dans la variable globale CLAMP.

1.5 Changement de fitness

Toutes les méthodes de changement de fitness sont dans le fichier FitnessChange.m.
Elles sont stockées dans la variable globale FITNESS_CHANGE.

Les méthodes de changment de fitness suivantes ont été implémentées :

- échelle linéaire,
- troncature sigma, avec un paramètre de contrôle c dans $[1, 5]$.

Dans le cas d'une maximisation où des fitness négatives sont produites, l'ensemble des fitness est translatée de sorte à ce que la fitness minimale soit 0. Cela revient à maximiser : $f'(x) = f(x) - min(f(x))$.

Dans le cas d'une minimisation, la méthode du transfert de fitness est utilisée.
C'est à dire que minimiser $f(x)$ revient à maximiser $f'(x) = max(f(x)) - f(x)$.

1.6 Classement

Toutes les méthodes de classement sont dans le fichier Ranking.m.
Elles sont stockées dans la variable globale RANKING.

Les méthodes de classement suivantes ont été implémentées :

- linéaire, avec un paramètre de contrôle α (β est déduit),
- linéaire (2ème méthode), avec un paramètre de contrôle t dans $[0, 1]$ (pour calculer r ; q est déduit),
- non-linéaire, avec un paramètre de contrôle α dans $]0, 1[$.

1.7 Stratégie *Steady State*

Un paramètre λ permet d'utiliser la stratégie *steady state* ($\lambda = 1$ ou 2) ou non ($\lambda = -1$).

1.8 Critères d'arrêt

Toutes les méthodes d'arrêt sont dans le fichier StopCriteria.m.
Elles sont stockées dans la variable globale STOP_CRITERIA.

Les critères d'arrêt suivants ont été implémentés :

- time,
- threshold, avec deux paramètres de contrôle :
  - r, la relation entre la fitness et la limite t (>=, <=, ...),
  - t, la limite fixée.
- variance, avec un paramètre de contrôle v : l'algorithme s'arrête quand la variance de la fitness est inférieur ou égale à v,

- min-max ratio, avec un paramètre de contrôle r, l'algorithme s'arrête lorsque le rapport entre la valeur maximale de la fitness et sa valeur minimale est proche de r,
- mean-change rate, avec un paramètre de contrôle cr : l'algorithme s'arrête quand la différence entre la valeur moyenne de la fitness actuelle et la précédente est inférieur ou égale à cr,

N.B : Même si un critère d'arrêt différent du temps est défini, l'algorithme ne dépassera pas le nombre d'itérations maximum donné.

2 Benchmarks

2.1 Configuration

Dans les deux problèmes d'optimisation, nous utilisons la fonction donnée comme fonction de fitness.
Le maximum de la fonction de Rosenbrock négative est en $(1.733..., 3)$. La valeur associée est 2.0025.

Nous avons choisi un critère d'arrêt $f(x, y) > 2.002$.

Le minimum de la fonction de Griewank est en $(0, 0)$. La valeur associée est 0.

Nous avons choisi un critère d'arrêt $f(x, y) < 10^{-4}$.

G_{max} est fixé à 250.

En encodage réel, nous utilisons *boundary mutation* comme fonction de mutation.
Lorsque qu'une solution dépasse l'interval, nous lui assignons ses valeurs extrêmes (*default clamp*).

Les autres paramètres sont les paramètres par défaut de l'algorithme.
Chaque combinaison a été effectuée 50 fois.

2.2 Affichage

Pour chaque test, deux affichages sont produits :

- le logarithme (en base 10) du nombre d'itérations avant que l'algorithme ne s'arrête,

- le logarithme (en base 10) de l'erreur entre la meilleur valeur trouvée et le maximum possible de la fonction de fitness. L'erreur attendue (par rapport au critère d'arrêt) est représentée par la ligne verte. Les valeurs au dessus de cette ligne ne satisfont pas le critère d'arrêt.

2.3 Encodage réel

(Voir Figures 1 - 6 et 12 - 17)

Nous avons fait varier les paramètres suivants :

- N, le nombre d'individus : 100, 200, 500 et 1000,
- α, le paramètre de contrôle de la fonction de crossover BLX-α : 0.25, 0.5 et 0.75,
- k, le paramètre de contrôle de la fonction de sélection par tournoi : 2 et 5,
- P_C, la probabilité de crossover : 0.8 et 0.65,
- P_M, la probabilité de mutation : 0.001 et 0.01.

Nous avons ensuite comparé différentes fonctions de sélection (*wheel selection* et *stochastic universal sampling*).

2.3.1 Commun

On remarque que le nombre d'individus influe grandement sur le nombre d'itérations nécessaire pour satisfaire le critère d'arrêt : plus il est grand, plus l'algorithme est performant.
Plus k est grand, meilleur est l'algorithme.

2.3.2 Rosenbrock

On peut remarquer que pour une combinaison donnée, plus le paramètre α est grand, meilleur est l'algorithme.
Une probabilité de crossover modérée ($P_C = 0.65$) ainsi qu'une faible probabilité de mutation ($P_M = 0.01$) semble être préférables.
La méthode du tournoi avec $k = 5$ semble la meilleure fonction de sélection suivie de *wheel selection*, la sélection par tournoi avec $k = 2$ et finalement *stochastic universal sampling*.

2.3.3 Griewank

On peut remarquer que pour une combinaison donnée, plus le paramètre α est petit, meilleur est l'algorithme.
Une grande probabilité de crossover ($P_C = 0.8$) ainsi qu'une très faible probabilité de mutation ($P_M = 0.001$) semble être préférables.
La méthode du tournoi est la meilleure fonction de sélection suivie de *stochastic universal sampling* et de *wheel selection*.

2.4 Encodage binaire

(Voir Figures 7 - 8 et 18 - 19)

Nous avons comparé les performances de *multi-point crossover* (avec n : 1, 2 et 10) par rapport à *1-bit adaptation crossover* paramétré avec les fonctions de crossover suivantes :

- *multi-point crossover*, avec un paramètre n : 1 (*single-point crossover*), 2 (*two-point crossover*),
- *uniform crossover*, avec un paramètre α : 0.25, 0.5 et 0.75.

2.4.1 Commun

L'encodage binaire donne des résultats comparables à l'encodage réel. 1-BX semble être plus performant que *multi-point crossover*.

2.4.2 Rosenbrock

Les résultats de 1-BX sont parmis les meilleurs lorsque $N = 1000$ et ce de manière constante (même comparé aux méthodes en encodage réel précédentes).

2.4.3 Griewank

En utilisant *multi-point crossover*, un trop grand nombre n de points semble faire baisser les performances de l'algorithme.

3 Remarques générales

Il vaut mieux préférer l'encodage réel à l'encodage binaire si l'on privilégie le temps de calcul.

Multi-point crossover est la fonction de crossover la plus lente et son temps d'exécution dépend du nombre de points n.

Les sélections par tournoi sont plus intéressantes que les autres sélections au niveau temporel :

- avec la sélection par tournoi non-biaisée et $k = 2$, il faut $0.5s$ pour effectuer 1000 itérations avec 1000 individus,
- avec *wheel selection*, il faut $1.5s$.

L'utilisation de l'algorithme *steady state* avec un remplacement par valeur permet de garantir que le meilleur individu de l'itération i est égal ou meilleur que celui de l'itération $i - 1$.
Cependant, étant donné que la population ne change que très peu à chaque itération, il est assez rare de trouver un solution acceptable en peu d'itérations.
Par contre, il fonctionne très bien avec une très faible population. (Voir Figure 10)
Il semble aussi mieux fonctionner en utilisant un système de classement, bien qu'il faille tout de même compter en milliers d'itérations (ce qui correspond à $\simeq 0.25s$ par millier).
Le problème de Griewank semble moins adapté à son utilisation.
(Voir Figures 11 et 21)

On peut trouver une solution au problème de Rosenbrock en une dizaine d'itérations (et même moins) avec :

- $N = 100$,
- *unbiased tournament* avec un $k >= 4$,
- *blend crossover* avec α proche ou égale à 1,
- *boundary mutation*.

Cela prend $\simeq 10ms$.
(Voir Figure 9)

On peut trouver une solution au problème de Griewank en une vingtaine d'itérations (et même moins) avec :

- $N = 500$,
- *unbiased tournament* avec un $k = 2$,
- *blend crossover* avec α proche ou égale à 0,
- *non-uniform mutation* avec $b = 2$.

Cela prend $\simeq 15ms$.
(Voir Figure 20)

4 Figures

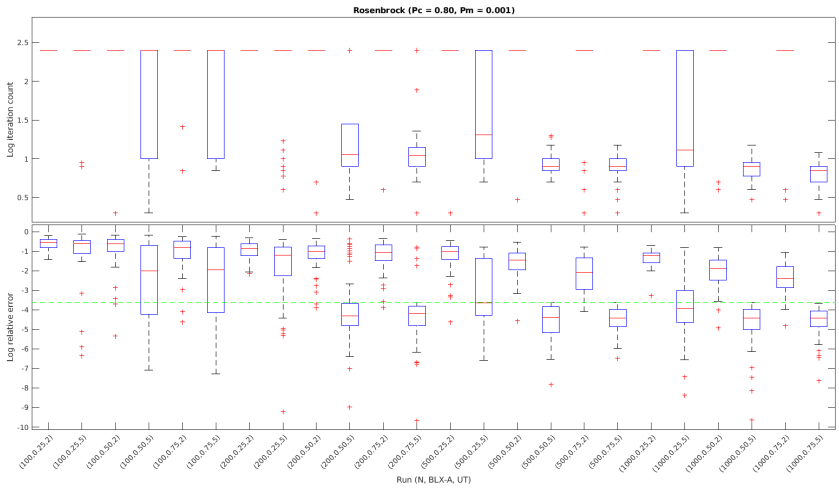

FIGURE 1 – Rosenbrock - BLX-α, Unbiased tournament, $P_C = 0.8$, $P_M = 0.001$

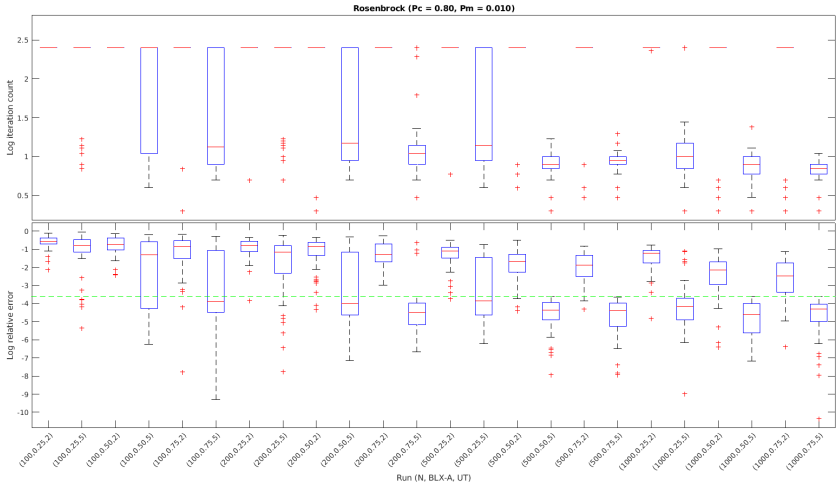

FIGURE 2 – Rosenbrock - BLX-α, Unbiased tournament, $P_C = 0.8$, $P_M = 0.01$

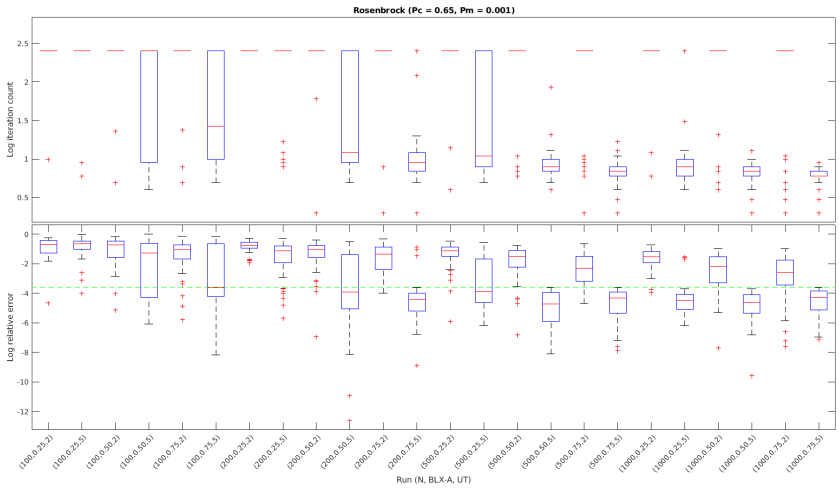

FIGURE 3 – Rosenbrock - BLX-α, Unbiased tournament, $P_C = 0.65$, $P_M = 0.001$

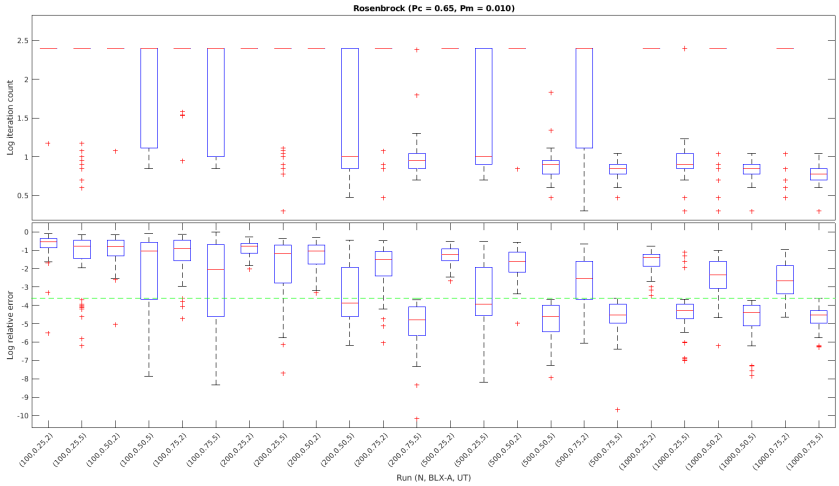

FIGURE 4 – Rosenbrock - BLX-α, Unbiased tournament, $P_C = 0.65$, $P_M = 0.01$

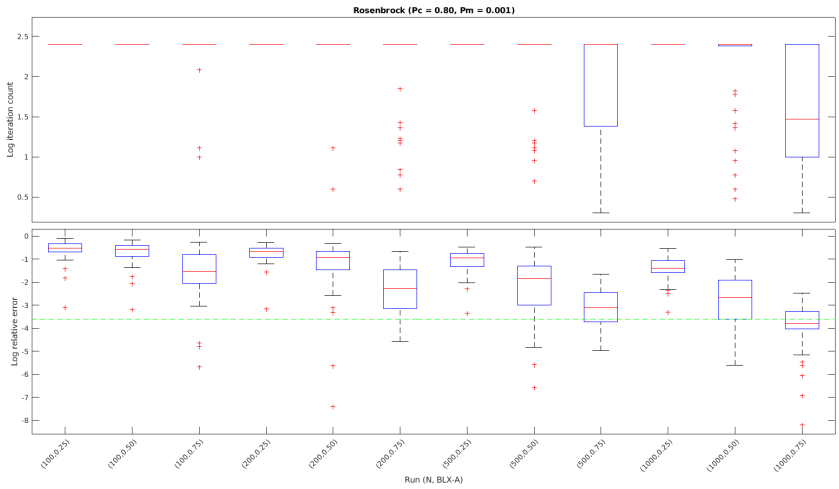

FIGURE 5 – Rosenbrock - BLX-α, Wheel

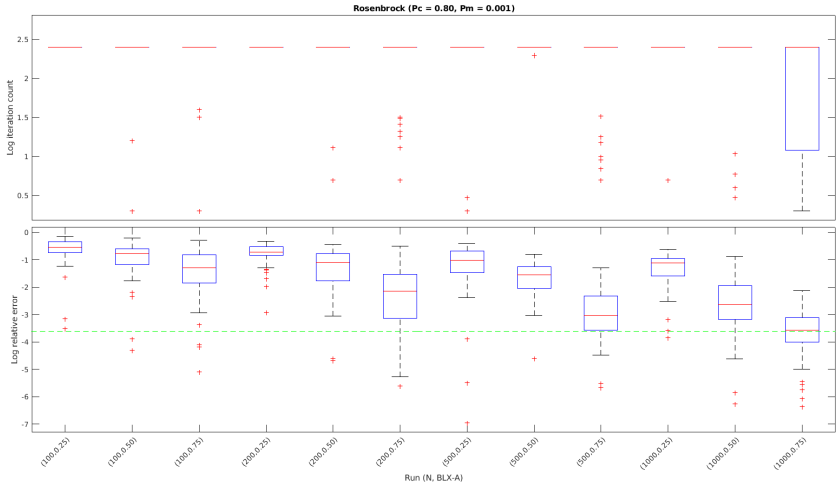

FIGURE 6 – Rosenbrock - BLX-α, Stochastic Universal Sampling

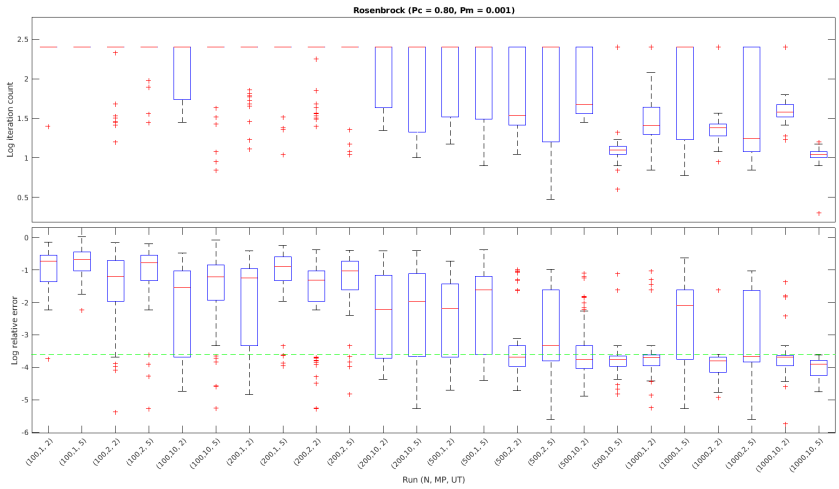

FIGURE 7 – Rosenbrock - Multi-point, Unbiased tournament

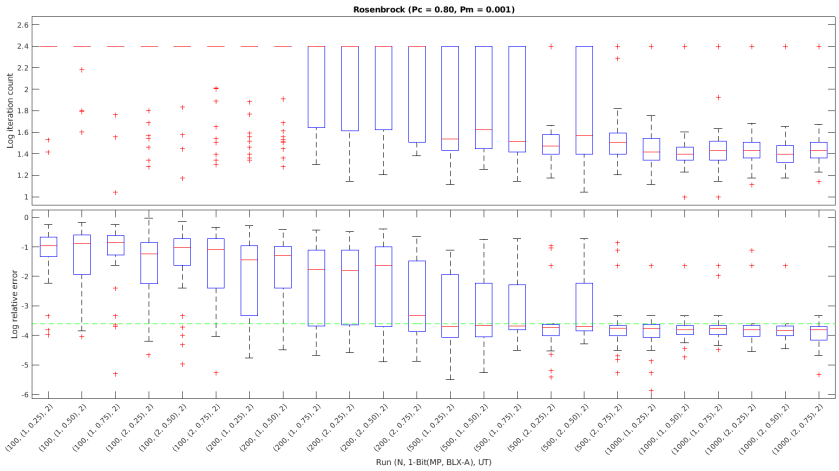

FIGURE 8 – Rosenbrock - 1-Bit Adaptation, Unbiased tournament

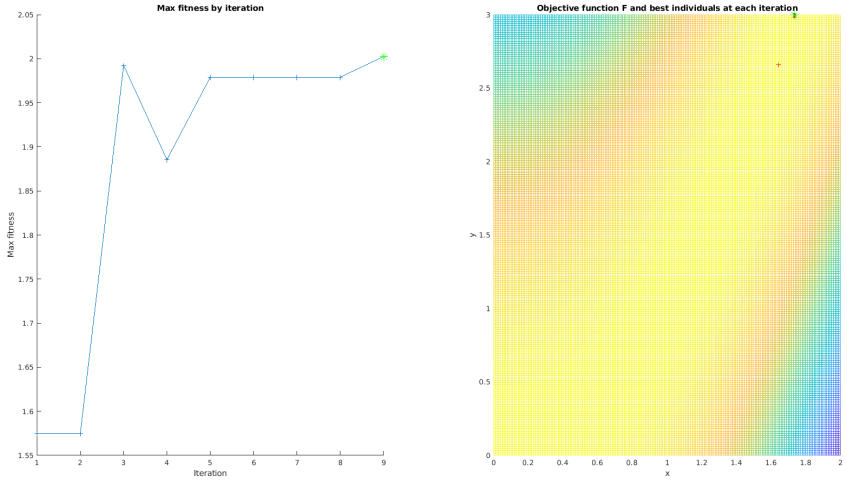

FIGURE 9 – Rosenbrock - $N = 100$, Unbiased tournament ($k = 4$), BLX-1, boundary mutation

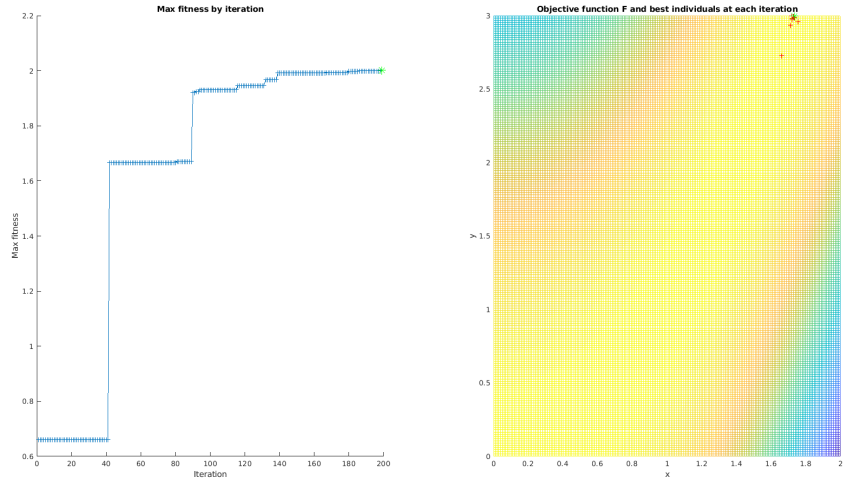

FIGURE 10 – Rosenbrock - $N = 10$, Steady State ($\lambda = 2$), BLX-1, non-linear ranking ($\alpha = 0.99$)

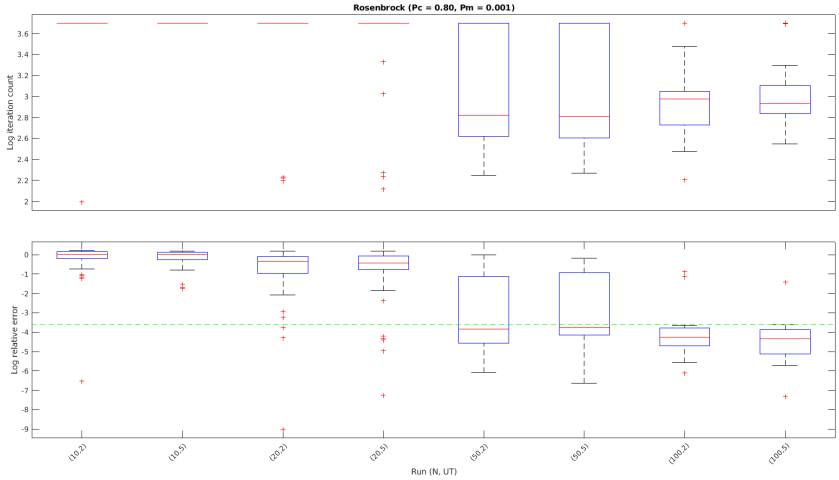

FIGURE 11 – Rosenbrock - Steady State ($\lambda = 2$), Unbiased tournament, BLX-1, non-linear ranking ($\alpha = 0.99$)

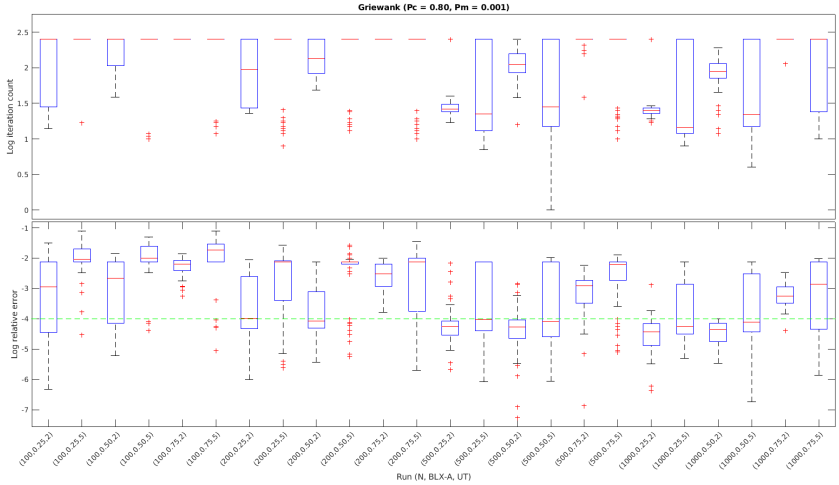

FIGURE 12 – Griewank - BLX-α, Unbiased tournament, $P_C = 0.8$, $P_M = 0.001$

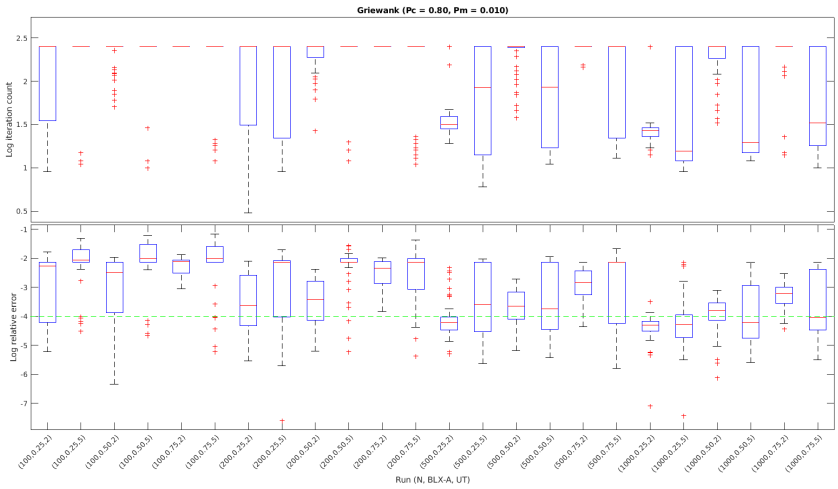

FIGURE 13 – Griewank - BLX-α, Unbiased tournament, $P_C = 0.8$, $P_M = 0.01$

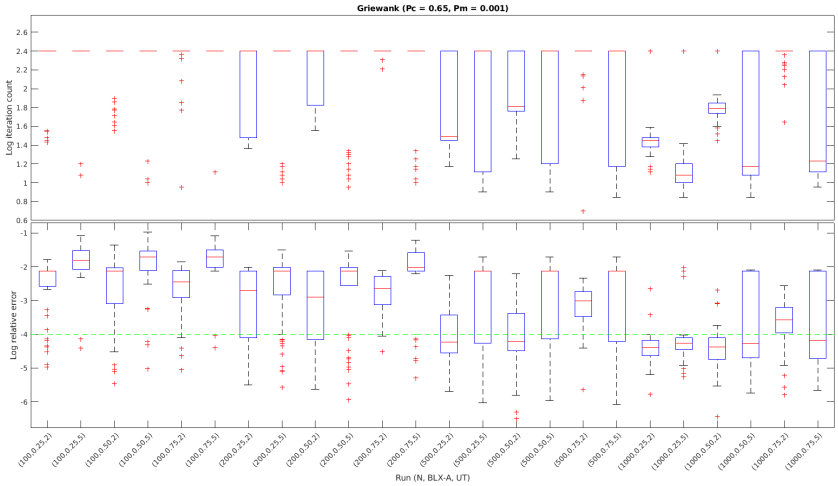

FIGURE 14 – Griewank - BLX-α, Unbiased tournament, $P_C = 0.65$, $P_M = 0.001$

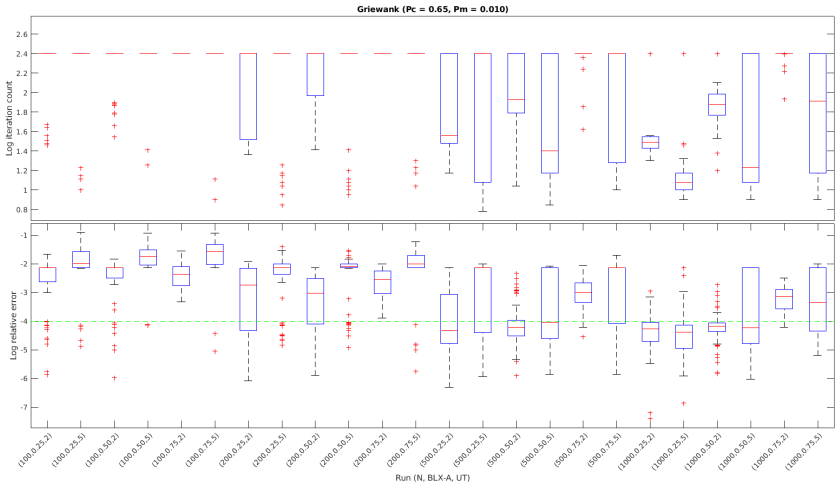

FIGURE 15 – Griewank - BLX-α, Unbiased tournament, $P_C = 0.65$, $P_M = 0.01$

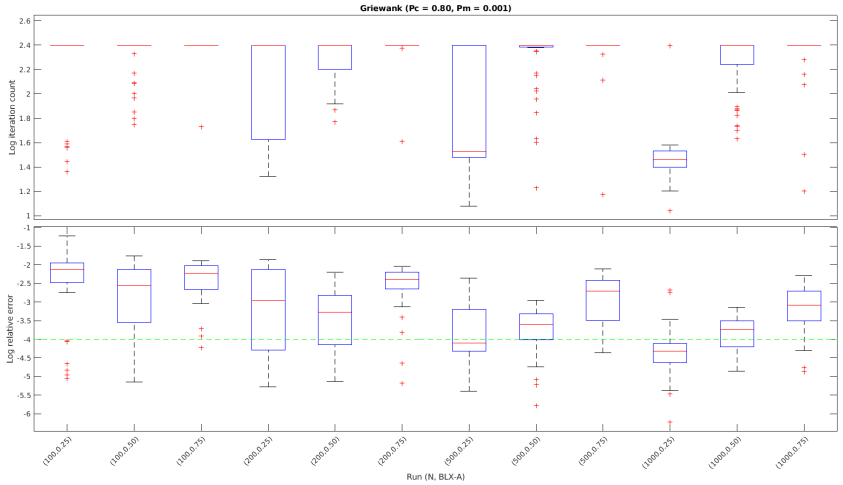

FIGURE 16 – Rosenbrock - BLX-α, Wheel

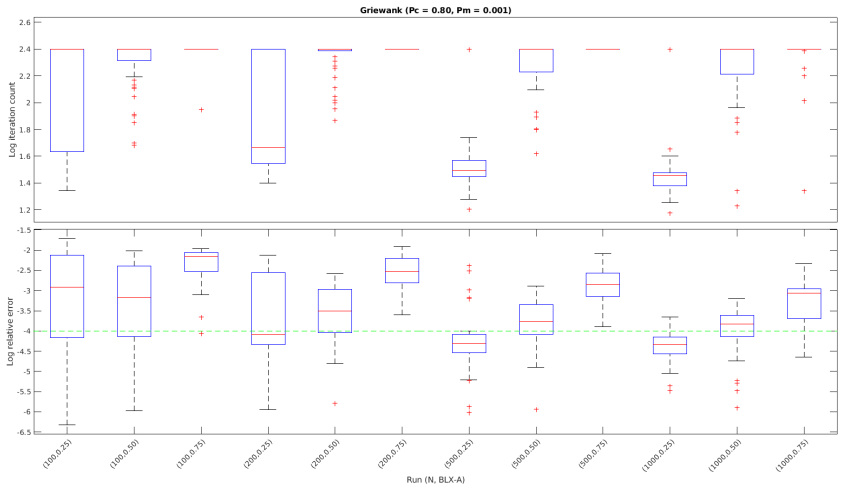

FIGURE 17 – Rosenbrock - BLX-α, Stochastic Universal Sampling

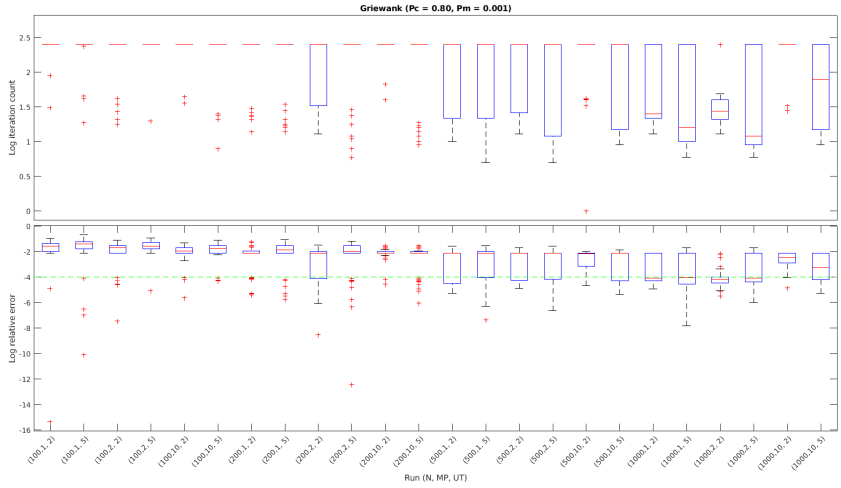

FIGURE 18 – Griewank - Multi-point, Unbiased tournament

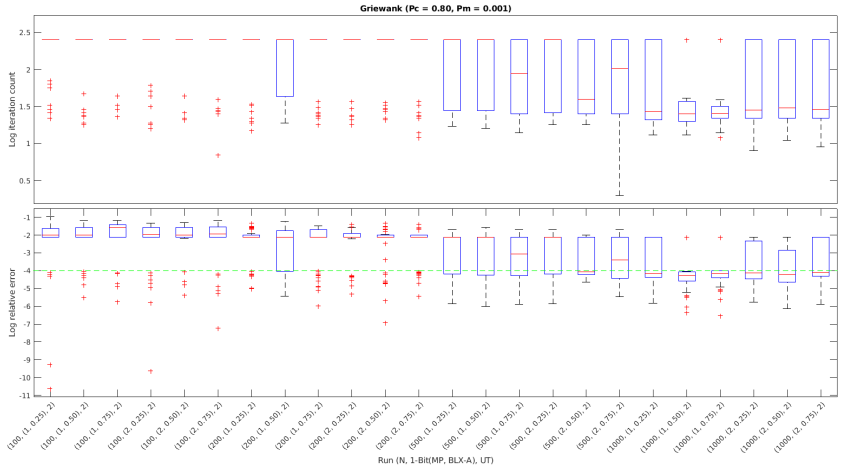

FIGURE 19 – Griewank - 1-Bit Adaptation, Unbiased tournament

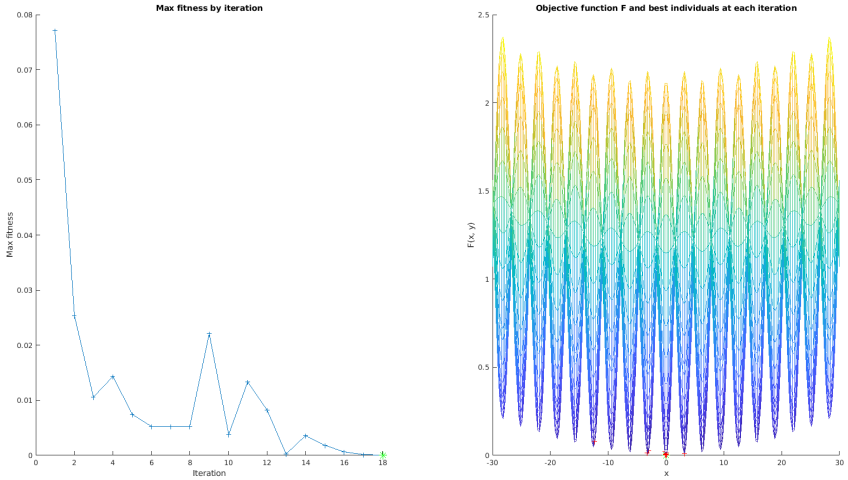

FIGURE 20 – Griewank - $N = 500$, Unbiased tournament ($k = 2$), BLX-0, Non-uniform mutation ($b = 2$), non-linear ranking ($\alpha = 0.99$)

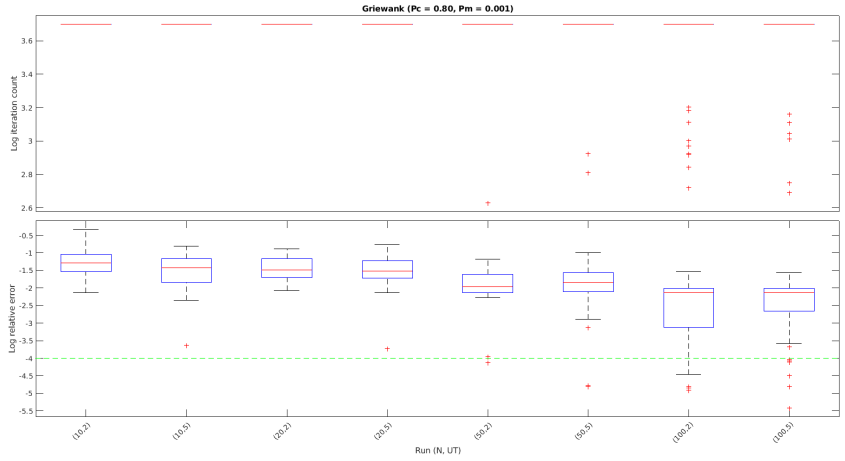

FIGURE 21 – Griewank - Steady State ($\lambda = 2$), Unbiased tournament, BLX-0, non-linear ranking ($\alpha = 0.99$)